UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/608,040	06/30/2003	Hajime Ogawa	2003_0866A	3923

513 7590 04/29/2008

WENDEROTH, LIND & PONACK, L.L.P.
2033 K STREET N. W.
SUITE 800
WASHINGTON, DC 20006-1021

EXAMINER	
WEI, ZHENG	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
04/29/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/608,040	OGAWA ET AL.	
	Examiner	Art Unit	
	ZHENG WEI	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *14 January 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-5,8-13,19-22,30,32,33,35,36,40,42,43 and 48-54* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-5, 8-13, 19-22, 30, 32, 33, 35, 36, 40, 42, 43, 48-54* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *30 June 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *11/07/2007, 03/21/2008*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Remarks

1. Applicants' remarks indicate that claims 30, 32, 33, 40, 42 and 43 have not been rejected in the previous final office action. After reviewing the office action, the Examiner notes that it is not a proper action and thus, the finality of that action is withdrawn.
2. This office action is in response to the amendment filed on 1/14/2008.
3. Claims 16-18, 23-28, 37-39 and 44-47 have been cancelled
4. Claims 1-3, 8-12, 19-22, 30, 32, 33, 42, 43 and 48-54 have been amended.
5. Claims 1-5, 8-13, 19-22, 30, 32, 33, 35, 36, 40, 42, 43 and 48-54 remain pending and have been examined.

Information Disclosure Statement

6. The information disclosure statements filed on 11/07/2007 and 3/21/2008 have been placed in the application file and the information referred to therein have already been considered.

Response to Arguments

7. Applicant's arguments filed on 1/14/2008, in particular on pages 18-30, have been fully considered but they are not persuasive. For example:

- At pages 18-21, section II "Claim Rejections under 35 U.S.C. § 101", the Applicants submit that Claims 5, 13, 23, 28 and 35 are statutory, because claims contains "functional descriptive material" (see for example, p.18-19"…with respect to source program, even though such programs may not be able to be directly executed by a computer, Applicants note that direct execution of a data structure or computer program is not required in order to be considered 'functional descriptive material'").

  However, the Examiner respectfully disagrees.

  First of all, the source program is not an executable computer program and thus it can not be executed and directs the computer to realize its functionalities. Secondly said source program is not "functional descriptive material". Although such source program/high-level language contains directives/flags, it is just an input to the compiler for generating executable code. Therefore, it is the compiler that imparts said functionality (not the source program/high-level language) when executed by the computer, can recognize said directives/flags in the source program and further direct the computer to generate optimized binary code. Therefore, 35 U.S.C. § 101 rejection is maintained.

- At page 21, bullet A of section III "Claim Rejections under 35 U.S.C. § 103(a)", the Applicants submit that "the ability to place variables at specific addresses, as disclosed in Popovic, does not in an way whatsoever correspond to the above noted features in claims of ahead address being indicated by a value

stored in a register, and a displacement that is within a range of the global memory region that can be accessed by on instruction, wherein the range is determined based on a type and size of an object."

However, the Examiner respectfully disagrees.

As Popovic disclosed, function "mem_alloc()" is used to allocate memory at a specific address (see for example, p.607, right column, last paragraph, "To enable programmers to manually allocated memory resource (to place variables at specific addresses)…"). Said specific address of the variable is "the head address" in claim 1 and the "size" of memory to be allocated is the "displacement" in claim 1 which is used to indicate the range of the allocated memory from the head address that can be accessed/referenced. The "range" in the claim is the same as the "size" which is used by Popovic to allocated the size of memory and such "range"/"size" is based on the type and size of object/variable. Because it is well known in the computer art that different type of variable has different size e.g. integer type, float type or string type variables. Therefore, allocating memory for different type of variable using the function "mem_alloc()" requires specifying variable size based on the variable type as the allocated memory range and a head address for a starting point of the allocated memory. In order to access the memory allocated by the Popovic, the pointer points to the head address has to be saved in Popovic's register (see for example, p.608, left column, "pointer registers").

- At the pages 23-24, bullet B of section III "Claim Rejections under 35 U.S.C. § 103(a)", the Applicants submit that Sun does not disclose or suggest the software pipelining itself and not performing the optimization by software pipelining of a specific loop processing.

  However, the Examiner respectfully disagrees.

  As Sun disclosed at page 12, "pragma pipeloop(n)" that is recognized by the compilation system is used to direct the compiler to pipeline number/all of successive iterations according to the "n" specified for the source code compilation. It should be noted that claim language does not define anything about "the software pipelining itself" as the Applicants argued. The Examiner reads the claim as the compiler apparatus (compilation system) can interpret the software (source code) and directive (pragma) by using a directive acquisition unit (recognized by compilation system) to perform/not to perform optimization during the source code compilation process. Sun also discloses that the pragma can be used to apply to specific loops which need to be pipelined (see for example, p.12, "The pragma applies to the next for loop within the current block…"). Therefore, it is obvious that only the loop with the specified pragma will be optimized in Sun and source code with other pragmas (except "#program pipeloop(n)") will not perform pipeline optimization as the Applicants argued in claim 8.

- At the page 24-28, bullet C of section III "Claim Rejections under 35 U.S.C. § 103(a)", the Applicants argue that Grantson only discloses that epilog can be

removed, but does not disclose that a directive is detected for performing optimization by software pipelining that removes a prolog portion and an epilog portion of a specific loop processing the source program as cited in claim 9.

The Examiner agrees with the Applicants and thus the rejection to claims 9-13, 37, 53 and 54 is withdrawn.

- At the pages 28-31, bullet E of section III "Claim Rejections under 35 U.S.C. § 103(a)", the Applicants submit that all the prior arts do not teach using the directive with different designation of the number of iterations to direct the compiler to perform optimization by loop unrolling.

  The Examiner's position is that as prior arts Stallman, PGI and Geva taught in previous claim 18 by using specified number of iterations for loop unrolling optimization, claims19, 48, 20, 49, 21, 50, 22 and 51 are only the different implementations which specified different number of iterations e.g. the iteration sets to 0, even number, odd number or equivalent to/more than the number of envelopment by the loop unrolling. Such implementations are obvious to the person in the art to provide user with different options and more flexible optimization choices based on method disclosed by Stallman, PGI and Geva.

Claim Rejections - 35 USC § 101

8. 35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

9. Claims 1-5, 8-13, 19-22, 30, 32-33, 35, 42-43, and 48-54 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Claims 1-4, 8-12, 19-22, 30, 32-33, 42-43 and 48-54:

These claims merely claim different "a computer-implemented compiler apparatuses that comprise different software modules/components e.g. “a directive acquisition unit”, “an optimization unit”…and thus such “computer-implemented compiler apparatus”[emphasis added] can be interpreted as software compiler program listings per se. The descriptions or expressions of the programs are not physical “things”. They are neither computer components nor statutory processes, as they are not “acts” being performed. Such claimed computer programs do not define any structural and functional interrelationships between the computer software and hardware components which permit the computer program’s functionality to be realized. Therefore, said computer-implemented computer programs (compilers) claimed as computer listings per se are nonstatutory. See M.P.E.P. 2106.01 (I)

Claims 5, 13, and 35: These claims claim the source programs, which are recorded on computer-readable recording medium, wherein the source program includes at least one of descriptions. However, (1) source programs is not executable by the computer hardware and can not direct the computer hardware to realize its functionalities; (2) said descriptions (directives) are just symbols or

flags in the source code as indications which are used by the compiler during compilation process. Actually, it is the compiler, not the source program that is capable to being executed by computer, processing said source code with the descriptions and further directing the computer to generate optimized binary code. Therefore said source program only including said descriptions are "Nonfunctional descriptive materials" and when nonfunctional descriptive materials are recorded on some computer-readable medium, in a computer or on electromagnetic carrier signals, they are not statutory since no requisite functionalities are present to satisfy the practical, useful application requirements. See M.P.E.P. 2106.01 (II)

Claim Rejections - 35 USC § 102

10. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

11. Claims 30, 32, 42, 33 and 43 are rejected under 35 U.S.C. 102(b) as being anticipated by Stallman (Richard M. Stallman, Using and Porting the GNU Compiler Collection for GCC 3.1)

Claim 30:

Stallman discloses a computer-implemented compiler apparatus having instructions stored thereon for causing a computer to translate a source program into a machine language program, said compiler apparatus comprising:

- a directive acquisition unit operable to acquire a directive for optimizing a machine language program to be generated (see for example, p.9, "Optimization options"); and
- an optimization unit operable to perform optimization by generating a sequence of machine language instructions following the acquired directive (see for example, see p.49, section 3.10, "Options that Control Optimization"),
- wherein the optimization unit performs optimization by allocating data in a memory region following a directive when the optimization unit acquires the directive on alignment of the array data to be allocated in a memory region (see for example, p.177, section 5.33 Specifying Attributes of Variables. "aligned (alignment)" and related description, also see example, p.178, line 7 and line 19, "short array[3] __attribute__ ((aligned)):" and related description)
- wherein the directive acquisition unit acquires a directive for alignment of array data of a special type together with a directive for translating the source program (see for example, p.10, line 3, "-O –O0 –O1 –O2 -O3 –Os" and related text), and

- the optimization unit allocates all the array data of the special type declared in the source program in the memory region so that its head address matches the alignment (see for example, p.177, section 5.33 Specifying Attributes of Variables. “aligned (alignment)” and related description, also see example, p.178, line 1-19, “short array[3] __attribute__ ((aligned)):” and related description).

Claim 32:

Stallman discloses a computer-implemented compiler apparatus having instructions stored thereon for causing a computer to translate a source program into a machine language program, said compiler apparatus comprising:

- a directive acquisition unit operable to acquire a directive for optimizing a machine language program to be generated (see for example, p.9, “Optimization options”); and
- an optimization unit operable to perform optimization by generating a sequence of machine language instructions following the acquired directive (see for example, see p.49, section 3.10, “Options that Control Optimization”),
- wherein the optimization unit performs optimization by allocating data in a memory region following a directive when the optimization unit acquires the directive on alignment of the array data to be allocated in a memory region (see for example, p.177, section 5.33 Specifying Attributes of Variables.

"aligned (alignment)" and related description, also see example, p.178, line 7 and line 19, "short array[3] __attribute__ ((aligned)):" and related description)

- wherein the directive acquisition unit detects designation of alignment of data that a pointer variable of argument shown by the name of a specific variable indicates in the source program (see for example, p.182, lines 16-20, "If you declare or use arrays of variable of an efficiently-aligned type…") and
- the optimization unit performs the optimization assuming that the data that is an object of designation detected by the directive acquisition unit is allocated in the memory region by the designated alignment (see for example, p.182, lines 16-20, "the compiler generates for these pointer arithmetic operations…").

Claim 42:

Stallman also discloses the compiler apparatus according to claim 32, wherein the optimization unit generates a pair instruction for transferring two or more kinds of data at the same time regarding a memory access instruction for accessing the data to be allocated in the memory region (see for example, p.181, lines 26-29, "use the ldd and std (doubleword load and store) instructions").

Claim 33:

Stallman discloses a computer-implemented compiler apparatus having instructions stored thereon for causing a computer to translate a source program into a machine language program, said compiler apparatus comprising:

- a directive acquisition unit operable to acquire a directive for optimizing a machine language program to be generated (see for example, p.9, "Optimization options"); and
- an optimization unit operable to perform optimization by generating a sequence of machine language instructions following the acquired directive (see for example, see p.49, section 3.10, "Options that Control Optimization"),
- wherein the optimization unit performs optimization by allocating data in a memory region following a directive when the optimization unit acquires the directive on alignment of the array data to be allocated in a memory region (see for example, p.177, section 5.33 Specifying Attributes of Variables. "aligned (alignment)" and related description, also see example, p.178, line 7 and line 19, "short array[3] __attribute__ ((aligned)):" and related description)
- wherein the directive acquisition unit detects designation of alignment of data that a local pointer variable shown by the name of a specific variable indicates in the source program (see for example, p.182, lines 16-20, "If you declare or use arrays of variable of an efficiently-aligned type…") and

- the optimization unit performs the optimization assuming that the data that is an object of designation detected by the directive acquisition unit is allocated in the memory region by the designated alignment (see for example, p.182, lines 16-20, "the compiler generates for these pointer arithmetic operations…").

Claim 43:

Stallman also discloses the compiler apparatus according to claim 33, wherein the optimization unit generates a pair instruction for transferring two or more kinds of data at the same time regarding a memory access instruction for accessing the data to be allocated in the memory region (see for example, p.181, lines 26-29, "use the ldd and std (doubleword load and store) instructions").

Claim Rejections - 35 USC § 103

12. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

13. Claims 1-5 and 36 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stallman (Richard M. Stallman, Using and Porting the GNU Compiler Collection for GCC 3.1) in view of Nakamura (Nakamura et al., Architecture and Compiler Co-Optimization for High Performance Computing) and further in view of Popovic (Popovic et al., A C Compiler Design Concept Used for MAS Family of Digital Signal Processors)

Claim 1:

Stallman discloses a computer-implemented compiler apparatus having instructions stored thereon for causing a computer to translate a source program into a machine language program, said compiler apparatus comprising:

- a directive acquisition unit operable to acquire a directive for optimizing a machine language program to be generated (see for example, p.9, "Optimization options") and
- an optimization unit operable to perform optimization by generating a sequence of machine language instructions following an acquired directive (see for example, see p.49, section 3.10, "Options that Control Optimization"),

But does not explicitly disclose, wherein the optimization unit performs the optimization by deciding array data allocated to a global memory region following a directive when the directive acquisition unit acquires the directive on the array data to be allocated to the global memory region. However, Nakamura in the same analogous art of compiler optimization discloses using directive to allocate

array in software controllable memory (see for example, p.52, section 3, Directive-Based Compiler, also see p.53, section 3.2 Example of Directives). But neither of them explicitly discloses, wherein the optimization unit performs the optimization by deciding array data allocated to a global memory region following a directive when the directive acquisition unit acquires the directive on the array data to be allocated to the global memory region. However, Popovic in the same analogous art of compiler optimization discloses

- using directive to allocate memory resource (see for example, p.607, section II, Compiler Structure, "To enable programmers to manually allocate memory resources (to place variables at specific addresses) a new pragma directive has been introduced. This directive is called mem_alloc")
- wherein the global memory region is specified by ahead address (org:address) and a displacement (size) (see for example, p.607, last paragraph, example code, "#pragma mem_alloc(org:address,.. size:4)")
- wherein the head address is indicated by a value stored in a register (see for example, p.607, last paragraph , example code, the value of "address")
- wherein the displacement is within a range of the global memory region that can be access by one instruction

Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to add this specific optimization option to Stallman's gcc compiler by using Popovic's method to allocate global memory resource for array as disclosed by Nakamura. One would have been motivated

to do so to improve the performance for high performance computing as suggested by Nakamura (see for example, p.51, section 2.2 Benefit of SCM). It also would have been obvious to one having ordinary skill in the art at the time the invention was make to understand that such compiler application can be run in a computing device to compile software source program to perform the function as addressed above.

Claim 2:

Stallman, Popovic and Nakamura disclose the compiler apparatus/device according to claim 1, Nakamura further discloses:

- wherein the directive acquisition unit acquires designation of a maximum data size of array data to be allocated to a global memory region together with a directive for translating the source program (see for example, p.52, Figure 2. directive "!$scm begin (<array_name>…)" and related text, also see, p.53, section Example of Directives"),

But none of them discloses

- the optimization unit, out of array data declared by the source program, allocates array data whose maximum data size does not exceed the maximum data size to a global memory region and array data whose maximum data size exceeds the maximum data size to a memory region out of the global memory region.

However, it is well known in the computer art that memory resource includes global/local memory. If the "pragma" requested array size excesses global memory size, it will allocate local memory next to the global memory address space. Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use directive to specify the specific memory location where the compiler allocates the array variable to according to the predefined parameter (maximum data size). One would have been motivated to do so to improve the performance for high performance computing as suggested by Nakamura (see for example, p.51, section 2.2 Benefit of SCM)

Claims 3-4:

Stallman, Popovic and Nakamura disclose the compiler apparatus/device according to claim 1, Nakamura further discloses:

- wherein the directive acquisition unit detects a directive for not allocating/ allocating specific array data to the global memory region in the source program (see for example, p.52, Figure 2. directive "!$scm begin (<array_name>…)" and related text, also see, p.53, section Example of Directives"), and
- the optimization unit allocates array data that are an object of a directive detected by the directive acquisition unit to a memory region out of the global memory region/global memory region (see for example, p.53, section Example of Directives")

Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to add this specific optimization option to Stallman's gcc compiler. One would have been motivated to do so to improve the performance for high performance computing as suggested by Nakamura (see for example, p.51, section 2.2 Benefit of SCM)

Claim 5:

Claim 5 is a software program product version of claimed apparatus discussed in claims 1-4 above, wherein all claimed limitations have been address and/or set forth above. Therefore, as the references teach all the limitation of claims 1-4, they also teach the limitations of claim 5. Thus, it also would have been obvious.

Claim 36:

Claim 36 is a software program product version of claimed apparatus discussed in claims 1-4 above, wherein all claimed limitations have been address and/or set forth above. Therefore, as the references teach all the limitation of claims 1-4, they also teach the limitations of claim 36. Thus, it also would have been obvious.

14. Claims 8 and 52 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stallman (Richard M. Stallman, Using and Porting the GNU Compiler Collection

for GCC 3.1) in view of Sun (Sun workshop compiler c 4.2 document: C user's guide)

Claims 8 and 52:

Stallman discloses a computer-implemented compiler apparatus having instructions stored thereon for causing a computer to translate a source program into a machine language program, said compiler apparatus comprising:

- a directive acquisition unit operable to acquire a directive for optimizing a machine language program to be generated (see for example, p.9, "Optimization options"); and
- an optimization unit operable to perform optimization by generating a sequence of machine language instructions following an acquired directive (see for example, see p.49, section 3.10, "Options that Control Optimization")

But Stallman does not explicitly disclose using directive to perform/not perform software pipelining optimization. However, Sun in the same analogous art of software compiler, discloses using directive for optimization on software pipelining (see for example, p.12, "#pragma pipeline(n)").

Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to understand that if the directive is specified other options except "pipeline(n)", certainly the directive detect unit will detect a directive other than pipelining and optimization unit will not perform pipelining optimization. It also would have been obvious to one having ordinary skill in the

art at the time the invention was make to understand that such compiler application can be run in a computing device to compile software source program to perform the function as addressed above.

15. Claims 9-10, 53 and 54 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stallman (Richard M. Stallman, Using and Porting the GNU Compiler Collection for GCC 3.1) in view of Sun (Sun workshop compiler c 4.2 document: C user's guide) in further view of Granston2 (US 2002/0112228)

Claims 9-10 and 53-54:

Stallman and Sun disclose the compiler apparatus according to claim 8, Stallman further discloses insert prolog and epilog portion to increase execution but neither of them discloses:

- wherein the directive acquisition unit detects a directive for performing the optimization by software pipelining that removes/does not remove a prolog portion and an epilog portion of a specific loop processing in the source program, and
- the optimization unit performs the optimization by software pipelining of loop processing that is an object of the directive detected by the directive acquisition unit whenever possible to remove the prolog portion and the epilog portion.

However, Granston2 in the same analogous art of software pipelining discloses a method to remove the epilog and prolog to reduce code size (see for example, paragraph [0036], "Because the code in the prolog and epilog is an exact copy of portions of the kernel, it may be possible to eliminate all or part of the prolog and epilog code"). It is also well known in the computer art that the compilation directive can be added to selectively optimize the specific loop. One would have been motivated to compile software program more flexible options.

16. Claims 11-13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stallman (Richard M. Stallman, Using and Porting the GNU Compiler Collection for GCC 3.1) in view of Sun (Sun workshop compiler c 4.2 document: C user's guide) in further view of Granston2 (US2002/0112228) in view of Granston (US6,892,380)

Claim 11:

Stallman and Sun disclose the compiler apparatus according to claim 8, but neither of them discloses:

- wherein the directive acquisition unit detects designation of the number of iterations of specific loop processing in the source program, and
- the optimization unit performs optimization of loop processing that is an object of the designation detected by the directive acquisition unit based on the designated number of iterations.

However, Granston in the same analogous art of software pipelining discloses a designation of the number of iterations of specific loop (see for example, col.1, lines 29-39, "n represents the number of desired iterations"). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to specifically define the number of iteration. One would have been motivated to do so to use this number compare with minimum required trip count to decide executing pipelined version or original version as indicated by Granston (see for example, col.3, lines 43-49, "if (n>= min required trip count")

Claim 12:

Stallman, Sun and Granston disclose the compiler apparatus according to claim 11, Granston further discloses:

- wherein the designation of the number of the iterations is the minimum number by which the loop processing is iterated (see for example, col.3, lines 43-49, "if (n>= min required trip count"), and
- the optimization unit performs the optimization by software pipelining when the minimum number is equivalent to or larger than the number of iterations that overlap by software pipelining (see for example, col.3, lines 43-49, "if (n>= min required trip count; pipelined version; else original version; endif" and relate description).

Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to specifically define the number of iteration as

the minimum iterated number. One would have been motivated to do so to use this number to generate multiversion code indicated by Granston (see for example, col.3, lines 50-51, "This technique is referred to as multiversion code generation)

Claim 13:

Claim 13 claims a computer source code that is written according to requirement of compiler apparatus discussed in claims 8-10 above. Because all claimed limitations of said compiler apparatus have been address and/or set forth above in claims 8-10. Therefore, it would have been obvious to perform the feature that the directive specified while using such compiler apparatus to compile said computer source code.

17. Claims 19-22 and 48-51 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stallman (Richard M. Stallman, Using and Porting the GNU Compiler Collection for GCC 3.1) in view of PGI (PGI Workstation User's Guide-9 Optimization Directive and Pragmas) and further in view of Geva (Robert Y. Geva, US 6,539,541)

Claim 19:

Stallman discloses a computer-implemented compiler apparatus having instructions stored thereon for causing a computer to translate a source program into a machine language program, said compiler apparatus comprising:

- a directive acquisition unit operable to acquire a directive for optimizing a machine language program to be generated (see for example, p.9, "Optimization options"); and
- an optimization unit operable to perform optimization by generating a sequence of machine language instructions following an acquired directive (see for example, see p.49, section 3.10, "Options that Control Optimization"),
- loop unrolling option(see for example, see p.49, section 3.10, "Options that Control Optimization", and also see p.55, lines 38-42, "-funroll-loops" and related description).

but does not explicitly disclose wherein the optimization unit performs optimization by loop unrolling following a directive when the directive acquisition unit acquires the directive on the optimization by loop unrolling.

However, PGI in the same analogous art of compiler software optimization using directive discloses adding pragmas to C and C++ specific to perform or not perform loop unrolling (see for example, section 9.4 Adding Pragmas to C and C++, Table 9-2 C/C++ Pragma Summary, "unroll" and "nounroll" and related description). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to also add directive to gcc compiler to selectively perform or not perform loop unrolling optimization. One would have been motivated to do so selectively perform or not perform loop unrolling optimization to specific loop instead of all software programs.

But neither of them discloses detecting a directive of designation of the number of iteration. However, Geva in the same analogous art of loop unrolling discloses the number of iterations defined by the directive (see for example, col.9, lines 37-38, "where the value of loop iterations is read by the program from an input file"). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to specify the number of iterations of specific loop processing in the source program. One would have been motivated to do so to ensure the number of iterations can be determined at compile time and further guarantee the loop unrolling can be performed as suggest by Geva (see for example, col.9, lines 31-42)

Geva further discloses: the optimization unit restrains generation of an escape code that is needed in the case of the number of the iterations being 0 when the minimum number is 1 or more (see for example, col.10, lines 9-10, "When the unrolled loop is a counted loop, there is no need to test for the exit condition inside the unrolled body.").

Claim 20:

As per Stallman, PGI and Geva disclosed above are incorporated, Geva further disclose the optimization unit performs the optimization by loop unrolling when the minimum number is equivalent to or more than the number of development by the loop unrolling (see for example, col.10, lines 6-9, "One such optimization

may be loop unrolling where a loop is unrolled 'n' times, such that 'n-1' additional copies of the loop body are made").

Claims 21 and 22:

As per Stallman, PGI and Geva disclosed above are incorporated, but neither of them explicitly discloses the number of iteration is even/odd number. However, Geva in the same analogous art of loop unrolling discloses the number of iterations defined by the directive (see for example, col.9, lines 37-38, "where the value of loop iterations is read by the program from an input file"). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to specify the number of iterations of specific loop processing in the source program. One would have been motivated to do so to ensure the number (even/odd number) of iterations can be determined at compile time and further guarantee the loop unrolling can be performed as suggest by Geva (see for example, col.9, lines 31-42).

Claims 48-51:

Claims 48-51 are other version of device claims performing the similar claimed method as in claims 19-22 addressed above, wherein all claimed limitation functions have been addressed and/or set forth above and certainly a computer system would need to run and/or practice such function steps disclosed by reference above. Thus, they also would have been obvious.

18. Claims 35 and 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stallman (Richard M. Stallman, Using and Porting the GNU Compiler Collection for GCC 3.1)

Claim 40:

Claim 40 is a software program product (GNU gcc Compiler) version of claimed apparatus in claims 30, 32, 33 and 42-43 above, wherein all claimed limitations have been address and/or set forth above. Therefore, as the references teach all the limitation of claims 30, 32, 33 and 42-43, they also teach the limitations of claim 40. Thus, it also would have been obvious.

Claim 35:

Claim 35 claims a computer source program/code that is written according to the requirement of compiler apparatus discussed in claims 30, 32 and 33 above. Because all claimed limitations of said compiler apparatus have been address and/or set forth above in claims 30, 32 and 33. Therefore, it would have been obvious to perform the feature that the directive specified while using such compiler apparatus to compile said computer source code.

Conclusion

19. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

20. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Zheng Wei whose telephone number is (571) 270-1059 and Fax number is (571) 270-2059. The examiner can normally be reached on Monday-Thursday 8:00-15:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature of relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571- 272-1000.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/ZW/

/Tuan Q. Dam/

Supervisory Patent Examiner, Art Unit 2192